Bachelor's Thesis (Academic Year 2021)

Traffic Matrix Generation for Quantum Internet Simulation

Keio University, Faculty of Environment and Information Studies
Nozomi Tanetani

卒業論文要旨 - 2021 年度 (令和 3 年度)

量子インターネットシミュレーションのためのトラフィック行列生成

量子インターネットとは、量子もつれを利用した新しいインターネットであり、現在のインターネットの弱点を補う技術として期待されている。具体的には、より強固な暗号アルゴリズム、コンセンサス問題の高速解決、分散量子計算による計算能力の向上などが挙げられる。しかし、実用化はまだまだ遠く、実験段階にある。

実用化する際に必要な要素の一つとしてトラフィックテストがある。これはネットワークのテストであり、ユーザーが利用する際にプロトコルなどが正しく動作するかどうかを確認するために行われる。トラフィックテストのためにはトラフィックデータが必要であり、それを用意するには、実データを使うか、もしくは確率的に生成する必要がある。量子インターネットの場合には実データが存在しないので、データを生成する必要がある。

本プロジェクトでは、量子インターネットにおける「トラフィック」を定義し、現行のインターネットにおいて主流である重力モデルを使ったトラフィックデータ生成を量子インターネットに適用した。また、これを量子インターネットシミュレータである QUISP に実装し、テストを行った。これにより、量子インターネットにおけるトラフィックテストの基盤を示した。

キーワード:
1. 量子インターネット, 2. トラフィック行列, 3. 重力モデル, 4. トラフィック生成

慶應義塾大学 環境情報学部 4 年
種谷 望

Abstract of Bachelor's Thesis - Academic Year 2021

Traffic Matrix Generation for Quantum Internet Simulation

The Quantum Internet is a new Internet that uses quantum entanglement, and is expected to be a technology that will make up for the weaknesses of the current Internet. It is expected to make up for the weaknesses of the current Internet, such as stronger cryptographic algorithms, faster resolution of consensus problems, and increased computing power through distributed quantum computation. However, practical applications are still far away and are still in the experimental stage.

One of the necessary elements for practical use is traffic testing. This is a test of the network, to make sure that protocols and other aspects work correctly when users use them. For traffic testing, we need traffic data, which can be either real data or generated stochastically. In the case of Quantum Internet, real data does not exist, so we need to generate data.

In this project, I defined "traffic" in the Quantum Internet, and applied the traffic data generation using the gravity model, which is the mainstream in the current Internet, to the Quantum Internet. I also implemented this model in QUISP, a Quantum Internet simulator, and tested it. In this way, It have been shown the basis for traffic testing in the Quantum Internet.
Keywords :
1. Quantum Internet, 2. Traffic Matrix, 3. Gravity Model, 4. Traffic Generation

Keio University, Faculty of Environment and Information Studies
Nozomi Tanetani

Contents

List of Figures

List of Tables

List of Code Listings

Chapter 1

Introduction

1.1 Background

The Quantum Internet is a new internet that uses quantum entanglement and is expected to be a technology that will make up for the weaknesses of the current Internet. Specifically, more robust cryptographic algorithms, faster solving of consensus problems and increased computational power through distributed quantum computation. However, the Quantum Internet has not yet reached the practical stage due to many hardware limitations. Nevertheless, many cryptographic algorithms and protocols are still being proposed so research activities on the Quantum Internet are flourishing.

In implementing the Quantum Internet, it is necessary to test the network. This is done to make sure that the protocols, etc. work properly for user use. In order to test the network, we need traffic data. To prepare traffic data, you need to use real data or generate it stochastically. However, since the Quantum Internet is not yet in practical use, it is not possible to use real data. Therefore, it needs to be generated stochastically. In the current Internet, traffic data is generated from the gravity model and the maximum entropy model, etc. The purpose of this paper is to establish a method of traffic data generation for the Quantum Internet.

1.2 Research Contribution

The main contribution of this project is the establishment of a basis for traffic matrix generation in the Quantum Internet. The traffic matrix generation method used in the current Internet has been applied to the Quantum Internet. Thereby, implemented traffic data generation for a Quantum Internet simulator to help simulate the Quantum Internet.

1.3 Thesis Structure

This thesis is constructed as follows.

Chapter 2 describes the fundamentals of quantum information and the Quantum Internet. In chapter 3, I will introduce the knowledge of the traffic in classical communications

and findings from previous studies. Chapter 4 details the problem definition of this research and the methods used to solve the problem. Chapter 5 describes how the method was implemented in a Quantum Internet simulator with actual code. Chapter 6 describes the evaluation of this research. It describes how to test the implemented functions. Chapter 7 provides a summary of this research and describes the future development of the research.

Chapter 2

Theory of Quantum Information

2.1 History of Quantum Information

2.2 Qubit

2.3 Multiple Qubit System

2.4 Quantum Gates

2.4.1 Single Qubit Gates

2.4.2 Controlled Gates

2.5 Superposition

2.6 Entanglement

2.6.1 Bell Pair

2.7 Quantum Teleportation

2.8 Quantum Network

2.8.1 Quantum Key Distribution

2.8.2 Classical Repeater Network

2.8.3 Quantum Repeater Network

2.8.4 Entanglement Swapping

2.8.5 Entanglement Purification

2.8.6 Quantum State Tomography

2.8.7 Quantum Internet Simulator

[1]

Chapter 3

Theory of Classical Teletraffic

3.1 Telephone Traffic

Traffic is a word originally used in the fields of transportation and trade to describe the volume of traffic and trade being transacted. Teletraffic means traffic of communication. The first type of teletraffic is a telephone. There are two main definitions of traffic in telephony, one is simply the number of calls and the other is the duration of calls. In some cases, the synergistic product of these two values is called traffic density, and this is referred to as traffic.[2]

3.2 Poisson Process

A Poisson Process is a model for a series of discrete event where the average time between events is known, but the exact timing of events is random. The arrival of an event is independent of the event before.[3] In terms of teletraffic, we know the amount of traffic in a unit of time, but we do not know at what interval it is communicated.

A Poisson Process meets the following criteria,[3]

1. Events are independent of each other. The occurrence of one event does not affect the probability another event will occur.

2. The average rate (events per time period) is constant.

3. Two events cannot occur at the same time.

The Poisson process can explain the characteristics of telephone traffic mathematically well. In particular, if we add an exponential distribution describing the time interval of events to the Poisson process, we can represent most of the actual telephone traffic. In fact, the beginning of teletraffic theory was based on empirical studies and measurements, but with the observation of the Poisson nature of call arrivals, Poisson processes and exponential distributions became universal laws describing telephone traffic, and the curiosity of actually measuring it diminished. This has allowed us to generate a steady stream of traffic, and many performance indicators can be accurately predicted. As a result, the telephone tends to be less blocked as a communication network.[4]

3.3 Acutual Traffic Pattern in IP Network

With the advent of the Internet, Internet traffic appeared in the field of teletraffic as well. Whereas telephone traffic is voice traffic, Internet traffic is data traffic. Therefore, Internet traffic is the amount of packets/data moving across a computer network at any given time.

While telephones occupy a single communication channel for communication, the Internet divides data into packets for transmission, and packets of other communications also flow in the communication channel. Therefore, the telephone is a centralized control system, while the Internet is a self-sustaining decentralized system. As a result, its overall behavior is very complex and difficult to understand. Since the Internet is a self-sustaining distributed system, data that exceeds the transfer limit may suddenly be sent over the network.A sudden increase in the amount of data is called a burst. One of the characteristics of Internet traffic is that, unlike telephone traffic, it cannot be explained by Poisson and exponential distributions.[5] This is because, as written earlier, the telephone and the Internet have different characteristics as communications.

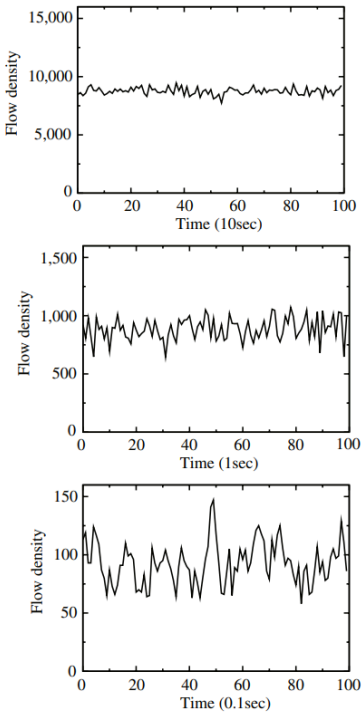

Figure 3.1: Exponential Traffic [1]

Figure. 3.1 is shows a graph of exponential traffic by timescale. As you can see, the exponential traffic is smoothed out as the timescale is increased.

[1] adapted from [5] Fig.2

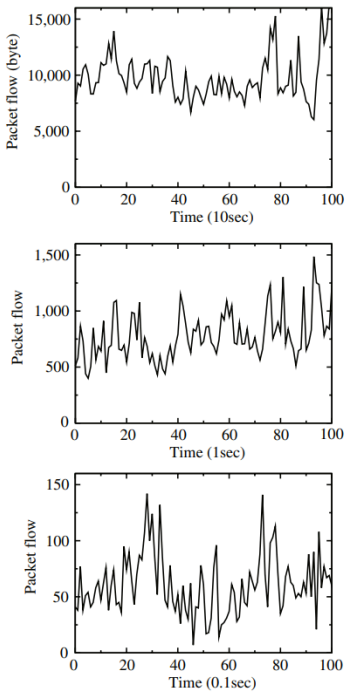

Figure 3.2: Actual Traffic [2]

Figure. 3.2 is shows a graph of actual traffic by timescale. As you can see, even with a larger timescale, the actual traffic is bumpy. This indicates that the traffic generation event occurrence is dependent on other events and that congestion can occur regardless of the timescale.

3.4 Self-similar Model

[2]adapted from [5] Fig.2
[3]adapted from [5] Fig.2

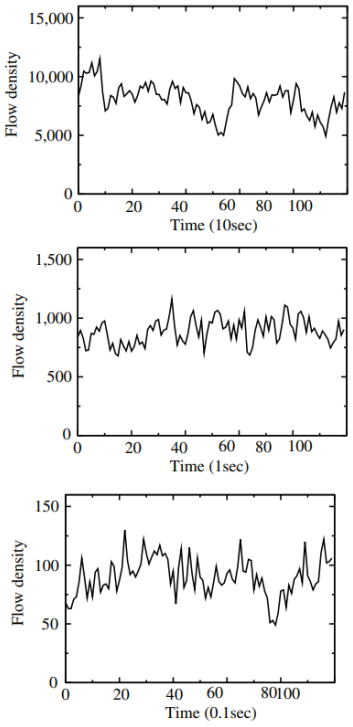

Figure 3.3: Self-Similar Traffic [3]

Table 3.1: The Features of Traffic [4]

	Telephone	Internet
Link usage	Occupied, continuous	Shared, discrete
Event occurrence	Poisson	Depends on other events
Remarks	Traffic can be expressed as Poisson, so traffic testing is easy.	Events depends on other events so traffic pattern is self-similar. We can't use Poisson to test IP network.

3.5 Traffic Matrix

A Traffic Matrix is a matrix giving the traffic volumes between origin and destination in a network and has tremendously potential utility for IP network capacity planning and management.[7]

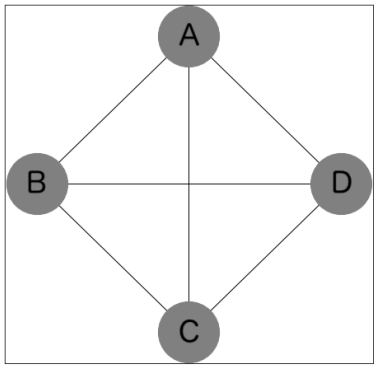

Figure 3.4: 4-node Network

Definition 1 (Traffic Matrix)**.** *Consider a simple four-node, all-coupled network as shown in Fig.3.4.* $T(i,j)$ *means the amount of traffic from point* j *to point* i*.*

$$\begin{bmatrix} T(A,A) & T(A,B) & T(A,C) & T(A,D) \\ T(B,A) & T(B,B) & T(B,C) & T(B,D) \\ T(C,A) & T(C,B) & T(C,C) & T(C,D) \\ T(D,A) & T(D,B) & T(D,C) & T(D,D) \end{bmatrix} \tag{3.1}$$

3.6 Gravity Model

Good for estimation and generation. Talk about estimation and synthesization.

[4] based on [6] Table.2

Chapter 4

Problem Definition and Proposal

4.1 Problem Definition

Create the base for traffic testing to make the Quantum Internet practical. Traffic tests in the Quantum Internet have yet to be devised. In this paper, I propose a definition of traffic in the Quantum Internet and a method to generate traffic data for traffic tests and show the base of traffic tests for the Quantum Internet.

4.2 Proposal

First, the traffic in the Quantum Internet should be defined. Traffic in the Quantum Internet is defined as one connection to one traffic. In other words, it does not matter how many bell pairs are shared in a single connection, or how long the connection takes. In the Quantum Internet, when a node wants to start a connection with another node, it sends a setup request to that node. Therefore, the number of setup requests sent by a node is the egress traffic, and the number of setup requests received by a node is the ingress traffic.

The generation of the traffic matrix for the Quantum Internet is done using the gravity model. This is identical to the method traffic is generated on the current Internet. The method of the current Internet will be applied to the Quantum Internet. For the gravity model type, the simplest SimpleGravityModel is used. The reason for this is that this is the first attempt to apply the gravity model to Quantum Internet traffic generation, and in addition, the gravity model itself can be made more complex at will later. The equation for Simple Gravity Model[8][9], which represents the total amount of traffic going from point j to point i, is as follows.

Definition 2 (Simple Gravity Model)**.**

$$T(i,j) = \frac{T_{ingress}(i) * T_{egress}(j)}{T_{total}} \tag{4.1}$$

$T(i,j)$ means the amount of traffic from point j to point i. $T_{ingress}(i)$ is ingress traffic at point i. $T_{egress}(j)$ is egress traffic at point j. T_{total} is the amount of overall traffic movement in the network.

The same equation for generating the traffic matrix is applied to the Quantum Internet. For that, need to clarify the definition of ingress traffic and egress traffic in Quantum Internet. The definitions of ingress traffic and egress traffic are described above. Ingress traffic is the number of connection setup requests received by a node.

Chapter 5

Implementation

5.1 Implementation

Code Listing 5.1: Gravity Generator Function

```
1  void Application::gravityGenerator() {
2    int traffic_ini = par("TrafficIni");
3    int res;
4    double res_all = 0;
5
6    cTopology *topo = new cTopology("topo");
7    topo->extractByParameter("nodeType", provider.getQNode()->par
       ("nodeType").str().c_str());
8
9    for (int i = 0; i < other_end_node_addresses.size(); i++) {
10     cTopology::Node *node = topo->getNode(i);
11     auto app_module = node->getModule()->getModuleByPath(".app"
         );
12     if (app_module == nullptr) throw cRuntimeError("app␣module␣
         not␣found");
13     res = app_module->par("TrafficRes");
14     res_all += res;
15   }
16   for (int i = 0; i < other_end_node_addresses.size(); i++) {
17     cTopology::Node *node = topo->getNode(i);
18     auto app_module = node->getModule()->getModuleByPath(".app"
         );
19     if (app_module == nullptr) throw cRuntimeError("app␣module␣
         not␣found");
20     res = app_module->par("TrafficRes");
21     traffic_matrix.push_back((int)round(traffic_ini * (res /
         res_all)));
22   }
23   int sum = std::accumulate(traffic_matrix.begin(),
        traffic_matrix.end(), 0);
24   while (sum != traffic_ini) {
25     std::vector<int>::iterator iter = std::max_element(
         traffic_matrix.begin(), traffic_matrix.end());
26     size_t max_index = std::distance(traffic_matrix.begin(),
         iter);
27     if (sum > traffic_ini) {
```

```
28        traffic_matrix[max_index] -= 1;
29      } else if (traffic_ini < sum) {
30        traffic_matrix[max_index] += 1;
31      }
32      sum = std::accumulate(traffic_matrix.begin(),
           traffic_matrix.end(), 0);
33    }
34
35    delete topo;
36  }
```

Code Listing 5.2: inside of initialize function

```
1  if (traffic_pattern == 3) {
2      gravityGenerator();
3      for (int i = 0; i < other_end_node_addresses.size(); i++) {
4          EV_INFO << traffic_matrix[i] << "␣requests␣will␣be␣sent
              ␣from␣" << my_address << "␣to␣" <<
              other_end_node_addresses[i] << "\n";
5          printf("%d␣requests␣will␣be␣sent␣from␣%d␣to␣%d\n",
              traffic_matrix[i], my_address,
              other_end_node_addresses[i]);
6      }
7      return;
8  }
```

Chapter 6

Evaluation

6.1 Evaluation

In this chapter, the verification of the correctness of the actual implemented code is described. The verification method is based on the following two procedures.

1. Confirm whether my traffic matrices generation calculation is correct
2. Verify that the generated traffic matrix is correct.

Chapter 7

Conclusion

7.1 Conclusion

In this research, I proposed a method for generating traffic matrices based on current Internet methods, and implemented it in a Quantum Internet simulator.

Bibliography

[1] Ryosuke Satoh, Michal Hajdušek, Naphan Benchasattabuse, Shota Nagayama, Kentaro Teramoto, Takaaki Matsuo, Sara Ayman Metwalli, Takahiko Satoh, Shigeya Suzuki, and Rodney Van Meter. Quisp: a quantum internet simulation package. *arXiv preprint arXiv:2112.07093*, 2021.

[2] 辻 舛二. **電話トラフィック理論とその応用**. 電子通信学会, 1954.

[3] Will Koehrsen. The poisson distribution and poisson process explained. `http://www.soumu.go.jp/menu_news/s-news/01tsushin02_02000072.html`, 2019. Accessed: 2021-12-18.

[4] Walter Willinger and Vern Paxson. Where mathematics meets the internet. *Notices of the AMS*, 45(8):961–970, 1998.

[5] 福田健介 et al. ネットワークトラフィックの自己相似性とその生成モデル. **情報処理**, 45(6):603–609, 2004.

[6] 上田浩, 奈須野裕, 岩谷幸雄, 五十嵐隆治, 木下哲男, et al. 確率過程による lan トラヒックのモデル化における一考察. **情報処理学会論文誌数理モデル化と応用** *(TOM)*, 48(SIG2 (TOM16)):167–174, 2007.

[7] Matthew Roughan. Internet traffic matrices. `https://roughan.info/project/traffic_matrix/`, 2017. Accessed: 2021-12-18.

[8] Yin Zhang, Matthew Roughan, Nick Duffield, and Albert Greenberg. Fast accurate computation of large-scale ip traffic matrices from link loads. *ACM SIGMETRICS Performance Evaluation Review*, 31(1):206–217, 2003.

[9] Matthew Roughan. How i learned to stop worrying and love traffic matrices. `https://roughan.info/talks/tma_summer_school.pdf`, 2016. Accessed: 2021-12-22.

[10] University of TARTU Institute of Computer Science. Gravity models. `https://courses.cs.ut.ee/2011/graphmining/Main/GravityModels`, 2011. Accessed: 2021-12-22.

[11] Nabili Benameur and JW Roberts. Traffic matrix inference in ip networks. *Networks and Spatial Economics*, 4(1):103–114, 2004.

[12] Matthew Roughan. Simplifying the synthesis of internet traffic matrices. *ACM SIGCOMM Computer Communication Review*, 35(5):93–96, 2005.

[13] Bernard Fortz, Jennifer Rexford, and Mikkel Thorup. Traffic engineering with traditional ip routing protocols. *IEEE communications Magazine*, 40(10):118–124, 2002.

[14] Vern Paxson and Sally Floyd. Wide area traffic: the failure of poisson modeling. *IEEE/ACM Transactions on networking*, 3(3):226–244, 1995.

[15] Matthew Roughan, Mikkel Thorup, and Yin Zhang. Traffic engineering with estimated traffic matrices. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, pages 248–258, 2003.

[16] Matthew Roughan, Albert Greenberg, Charles Kalmanek, Michael Rumsewicz, Jennifer Yates, and Yin Zhang. Experience in measuring internet backbone traffic variability: Models metrics, measurements and meaning. In *Teletraffic Science and Engineering*, volume 5, pages 379–388. Elsevier, 2003.

[17] Takaaki Matsuo, Clément Durand, and Rodney Van Meter. Quantum link bootstrapping using a ruleset-based communication protocol. *Physical Review A*, 100(5):052320, 2019.

[18] Takaaki Matsuo, Takahiko Satoh, Shota Nagayama, and Rodney Van Meter. Analysis of measurement-based quantum network coding over repeater networks under noisy conditions. *Physical Review A*, 97(6):062328, 2018.

[19] Takaaki Matsuo. Simulation of a dynamic, ruleset-based quantum network. Master's thesis, Keio University, Graduate School of Media and Governance, 2019.

[20] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

Appendix A

Appendix

A.1 Gravity Model Sample Code with Python

Code Listing A.1: Generate Simple Network with Python

```
#Author: Nozomi Tanetani
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt

def graph_generator(n, p, seed):
  g = nx.random_graphs.fast_gnp_random_graph(n, p, seed,
     directed = True) #Generate a random graph
  g = g.to_undirected()

  #Initialize a number of traffic per node as 0.
  for i in g.nodes():
    g.nodes[i]['in'] = 0
    g.nodes[i]['out'] = 0

  #Store a random number into number of traffic variable.
  for i in g.nodes():
    g.nodes[i]['in'] = np.random.randint(n*100,n*500)
    tmp = g.nodes[i]['in']
    for j in g.nodes():
      if i is not j:
        if j is n-1:
          g.nodes[j]['out'] += tmp
        else:
          rand_tmp = np.random.randint(0,tmp)
          g.nodes[j]['out'] += rand_tmp
          tmp = tmp - rand_tmp
  return g

g = graph_generator(4, 1, 13)

#Display a generated graph
plt.figure(figsize=(8,8))

pos = nx.circular_layout(g)
```

```
nx.draw_networkx_nodes(g, pos, node_size=5500, node_color='gray')
nx.draw_networkx_edges(g, pos)
nx.draw_networkx_labels(g, pos, font_size=20)

#Display
labeldict = nx.get_node_attributes(g,'in')
for i in g.nodes():
  labeldict[i] = '\n\n' + str(labeldict[i])
nx.draw_networkx_labels(g,pos,labels=labeldict,font_size=12, font_color='#FFAAAA')

labeldict = nx.get_node_attributes(g,'out')
for i in g.nodes():
  labeldict[i] = '\n\n\n\n' + str(labeldict[i])
nx.draw_networkx_labels(g,pos,labels=labeldict,font_size=12, font_color='#AAFFFF')
print("red number is total traffic that enters the node.")
print("blue number is total traffic that exits the node.")
plt.show()
```

Code Listing A.2: Gravity Generator Function with Python

```
def gravity_generator(graph):
    tm = np.zeros((n,n))

    for i in range(len(tm)):
        for j in range(len(tm[i])):
        if i is not j:
            tmp = 0
            for k in range(n):
            if k is not i:
                tmp += graph.nodes[k]['out']
            tm[i][j] = int(np.round(graph.nodes()[i]['in'] * (graph.nodes[j]['out'] / tmp))) # Slide Page 15
    for i in range(len(tm)):
        if sum(tm[i]) - graph.nodes()[i]['in'] == 1:
        tm[i][np.argmax(tm[i])] -= 1
        elif graph.nodes()[i]['in'] - sum(tm[i]) == 1:
        tm[i][np.argmax(tm[i])] += 1
    return tm

#Create a traffic matrix
tm = gravity_generator(g)
print(tm)
```